Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on February 10th 2011 has been entered.

2. Applicant's request for continued examination and response dated February 10th 2011, responding to the October 28th 2010, Final Office Action provided in the rejection of claims 1-2, 4-7, 9-10 and 12-20. Claims 1, 6, 10 and 12 have been amended. . Claims 1-2, 4-7, 9-10 are pending in this application and which have been fully considered by the examiner.

Prior Art's Arguments – Rejections

3. Applicant's arguments filed on February 10th 2011, in particular pages 10-12, have been fully considered but they are not persuasive. For example:

On pages 10, Applicant contends *Noble* does not disclose a library having complexity evaluations functions. However, Examiner strongly disagrees. (For example, *Noble* page 214-215, a suite of metrics are applied to the GUI in order to determine the overall complexity of a GUI. The suite is collection of complexity functions (as can be further seen by definitions in section 3.1-3.3 where task concordance is computed as the rank order correlation given by Kendall's τ. Layout Uniformity is computed using the function:

$$LU = 100 * \left(1 - \frac{(N_h + N_w + N_t + N_l + N_b + N_r) - M}{6 * N_x - M}\right)$$

Visual coherence is computed using the function.

$$VC = 100 * \left(\frac{\sum_{\forall i} G_i}{\sum_{\forall i} N_i * (N_i - 1)/2} \right)$$

with $G_i = \sum_{\forall j (i \neq j)} R_{i,j}$

The suite of metrics is thereby a library (i.e. a collection of functions) that evaluate to numerical results. Each of the functions in the suite evaluates different aspects of a user interface. The evaluation metrics of aspects of the interfaces include "task concordance" "layout uniformity" "visual coherence" and "interactive visualization." These evaluation functions represent various intricacies of the interface.

Responding to Applicant's assertion that "Noble teaches away from displaying a numerical value for each design element," Examiner respectfully disagrees. When reading the one paragraph Applicant cites out of context it appears that Noble is teaching away from using his very own disclosure (i.e. from using the developed algorithms that evaluate to numerical results). However, when reading the entire section it becomes clear that Noble is in fact talking about a method of interactive visualization that should not be manipulated using just the numerical results displayed. Noble describes this in terms of two interface design systems, Hydra and AIDE which make this very mistake. Noble then goes on to describe an interactive editing method allowing for "simultaneous feedback in multiple dimensions." Thus, Noble does not teach away from display numerical results.

Applicant's arguments with respect to displaying an aggregated complexity result for each device class is moot in light of the further explained rejection (i.e. it is moot as *Comber* is no longer relied upon for the teaching of the limitation). Examiner points out that Examiner relies on the teachings of *Parker* for the device class specific representation of the user interfaces. Furthermore, *Comber* discloses the aggregation of complexity values per user

interface. Therefore, the combination of Noble, Parker and Comber together disclose displaying an aggregated complexity result for each device class as further described in the rejection below.

Abstract

4. Applicant's abstract is objected to as being in improper form. Applicant has appeared to submit the face of the corresponding PCT application as the abstract. Applicant is required to submit a new abstract on a separate sheet.

Applicant is reminded of the proper content of an abstract of the disclosure:

A patent abstract is a concise statement of the technical disclosure of the patent and should include that which is new in the art to which the invention pertains. If the patent is of a basic nature, the entire technical disclosure may be new in the art, and the abstract should be directed to the entire disclosure. If the patent is in the nature of an improvement in an old apparatus, process, product, or composition, the abstract should include the technical disclosure of the improvement. In certain patents, particularly those for compounds and compositions, wherein the process for making and/or the use thereof are not obvious, the abstract should set forth a process for making and/or use thereof. If the new technical disclosure involves modifications or alternatives, the abstract should mention by way of example the preferred modification or alternative.

The abstract should not refer to purported merits or speculative applications of the invention and should not compare the invention with the prior art.

Appropriate action is required.

Claim Objections

5. Claims 13-14 and 17-18 are objected to for lack of antecedent basis. The claims refer to the limitation "child nodes" (claim 13 line 2 and claim 17 line 2) and "parent nodes" (claim 13 and 17 line 2, claim 14 and 18 line 3) which has not been recited in any of the preceding claims. A hierarchy is simply an ordering of a set of objects. However, a hierarchy does not imply the existence of a "node" as Applicant argues. Therefore, while a hierarchy does imply an object with a higher rank then another in a set ordering (i.e. parents and children to use Applicant's terminology), the hierarchy does imply the existence of a nodal structure. The objection to claims 13-14 and 17-18 is hereby upheld.

Claim Rejections - 35 USC § 101

6. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claim 1 is rejected under 35 U.S.C. 101 as directed to non-statutory subject matter of software, *per se*. The claim lacks the necessary physical articles or objects to constitute a machine or manufacture within the meaning of 35 U.S.C. 101. It is clearly not a series of steps or acts to be a process nor are they a combination of chemical compounds to be a composition of matter. As such, they fail to fall within a statutory category. It is at best, function descriptive material *per se*.

In this case, applicant has claimed a "a complexity indicator" which contains a library an aggregator and a complexity display; this implies that Applicant is claiming a system of software, per se, lacking the hardware necessary to realize any of the underlying functionality. Therefore, claim 1 is directed to non-statutory subject matter as computer programs, per se, i.e. the descriptions or expressions of the programs, are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer, which permit the computer program's functionality to be realized.

Claims 2, 4-5 are rejected under 35 U.S.C. 101 as non-statutory for at least the reason stated above. Claims 2, 4-5 are depended on claim 1; however, they do not add any feature or subject matter that would solve any of the non-statutory deficiencies of claim 1.

Examiner suggests Applicant amend claim 1 to include one of the following:

"A complexity indicator having a memory storing instructions …"

Or

"A complexity indicator having a processor for executing instructions…"

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-2, 4-7, 9-10, and 12-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over James Noble et al. ("Interactive Design Metric Visualization: Visual Metric Support for User Interface Design" IEEE 1996) (hereinafter Noble) in view of Parker et al. (US 5,600,789) (hereinafter Parker) and in view of Tim Comber and John Maltby ("Investigating Layout Complexity" March 2003) (hereinafter Comber).

As per claim 1, Noble discloses a computer complexity indicator having instructions to evaluate the complexity of a user interface that has device class specific representations, [each device class specific representation referring to a respective device class] and having a respective layout component hierarchy; (For example, abstract, interactive metric visualization is a novel approach providing complex, multi-dimensional feedback on the effects of layout changes in user interface designs; for example, page 213 section 2.1, the layout contains a hierarchy of components that a visualization IDE such as IBM's VisualAge can display. A device class specific representation is a specific user interface. The example of IMB's VisualAge as well as the design implemented by the prior art is intended for use with all user interfaces).

the complexity indicator comprising: a library having complexity evaluation functions to determine complexity values of layout components of the respective layout component hierarchies, where each complexity evaluation function is specific to the layout component [and the device class] to which it is applied the complexity values being numerical values; (For example, page 213-215 sections 2.2-3.3, a suite (library) of metrics (complexity evaluation functions) are used to determine the complexity values of a corresponding user interface. The

complexity evolution functions are associated with each of the components to which the functions are applied (see also figures 1-4). The functions evaluate to numerical values).

and an aggregator to aggregate, using one or more processors, [the complexity values into a single complexity value for each device class] according to the corresponding layout component hierarchy of the respective [device class specific] representation (For example, figure 4, the results are aggregated together to show the relationships between the components (hierarchy) as well as the complexity of the interface layout. The display pertains to the specific user interface currently under observation but is not limited to just one user interface (i.e. the tool may execute multiple times with each resulting execution associated with a different UI)).

Noble does not expressly disclose each device class specific representation referring to a respective device class.

However, Parker discloses each device class specific representation referring to a respective device class (For example, figure 13 and 15, each operating environment and computer has a separate user interface design that is unique to that respective operating environment or computer).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble and extend it to multiple test multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

Noble and Parker do not expressly disclose the complexity values into a single complexity value for each [device class] and a complexity display to visually present the aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value.

However, Comber discloses the complexity values into a single complexity value for each device class and complexity display to visually present the aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value (For example, page 225, table 8 shows the aggregated complexity value for a device class as a numerical value. Table 2 shows an aggregated complexity value per user interface executing on the device class. The Total C column is the aggregated complexity value).

Noble, Parker and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system that determines complexity of each user interface, which executable on a different device class, numerically as described by Noble and Parker extend it to define complexity in terms of numerical, graph-able values per user interface as taught by Comber because it would provide with an efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

As per claim 2, Noble discloses the complexity indicator of claim 1, further comprising: a transformer to transform the layout component hierarchy of each representation into a corresponding complexity evaluation hierarchy so that the association of each evaluation

function with its respective layout components is redirected through the corresponding component of the respective complexity evaluation hierarchy and the evaluation function is applied to the corresponding component of the respective complexity evaluation hierarchy (For example, figures 2-4, the layout hierarchy is transformed into a complexity evaluation hierarchy (as seen in the figures) where the different lines reflect the metrics (complexity evaluation functions) applied to the different layout components of the complexity evaluation hierarchy).

As per claim 4, Noble discloses the complexity indicator of claim 1, wherein the complexity display [has a drill down portion] to visualize complexity values of layout components related to a selected [device class] (For example, figure 4, the figure depicts a complexity evaluation hierarchy based on the selected (currently evaluated) user interface).

Noble does not expressly disclose a drill down portion to visualize complexity values of layout components related to a selected device class.

However, Parker discloses a drill down portion to visualize complexity values of layout components related to a selected device class (For example, figure 13 and 15, the display of the evaluation of the user interface is based on the selected device (WINDOWS or MAC operating environments (figure 13) or computer 1 or 2 (figure 14)). All devices tested allowing for a selection of one particular devices and then viewing results (i.e. drill down)).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble and extend it to multiple test

multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

As per claim 5, Noble discloses the complexity indicator of claim 4 in combination with a tree-based outline editor to generate an outline views of the representations that corresponds to the selected device class configured to highlight a layout component that is selected in the complexity display for drill down purposes (For example, figure 3 and page 214 section 3.1, task concordance is a measure of the fit between the expected frequency of various tasks and their relative difficulty using a given interface design. TC is computed as the rank order correlation between tasks ranked by operational difficulty (e.g. steps or path length) and by anticipated frequency of use. The figure 3 shows an interactive tree-based outline representation of TC which orders the difference tasks based on frequency and pathlength. The tree is based on a selected device class (UI that is currently selected for evaluation).

As per claim 6, Noble discloses a method for complexity evaluation of a user interface, comprising: (For example, abstract, interactive metric visualization is a novel approach providing complex, multi-dimensional feedback on the effects of layout changes in user interface designs).

determining complexity values of layout components of the [device class] specific representations by applying complexity evaluation functions that are specific to the respective layout components and the respective [device class to which the complexity evaluation function

is applied], the complexity values being numerical values; (For example, page 213-215 sections 2.2-3.3, a suite (library) of metrics (complexity evaluation functions) are used to determine the complexity values of a corresponding user interface. The complexity evolution functions are associated with each of the components to which the functions are applied (see also figures 1-4). The functions evaluate to numerical values).

and aggregating, using one or more processors, the complexity values [into a single complexity value for each device class] according to a corresponding layout component hierarchy of the respective [device class specific] representation (For example, figure 4, the results are aggregated together to show the relationships between the components (hierarchy) as well as the complexity of the interface layout. The display pertains to the specific user interface currently under observation but is not limited to just one user interface (i.e. the tool may execute multiple times with each resulting execution associated with a different UI)).

Noble does not expressly disclose device class to which the complexity evaluation function is applied and device class specific representations of the user interface, wherein each device class specific representation referring to a respective device class.

However, Parker discloses device class to which the complexity evaluation function is applied receiving device class specific representations of the user interface, wherein each device class specific representation referring to a respective device class; (For example, figure 13 and 15, each operating environment and computer has a separate user interface design that is unique to that respective operating environment or computer. Each user interface which is unique to a respective design class is tested).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble and extend it to multiple test multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

Noble and Parker do not expressly aggregating, using one or more processors, the complexity values into a single complexity value for each [device class] and disclose visually representing the aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value.

However, Comber discloses and aggregating, using one or more processors, the complexity values into a single complexity value for each [device class] visually representing the aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value (For example, page 225, table 8 shows the aggregated complexity value, visually represented, for each device class (Scr.1 - Scr.4) as a numerical value).

Noble, Parker and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system that determines complexity of each user interface, which executable on a different device class, numerically as described by Noble and Parker extend it to define complexity in terms of numerical, graph-able values per user interface as

taught by Comber because it would provide with an efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

As per claim 7, Noble discloses the method of claim 6, further comprising: transforming the layout component hierarchy of each representation into a corresponding complexity evaluation hierarchy so that the association of each evaluation function with its respective layout component is redirected through the corresponding component of the respective complexity evaluation hierarchy and the evaluation function is applied to the corresponding component of the respective complexity evaluation hierarchy (For example, figures 2-4, the layout hierarchy is transformed into a complexity evaluation hierarchy (as seen in the figures) where the different lines reflect the metrics (complexity evaluation functions) applied to the different layout components of the complexity evaluation hierarchy).

As per claim 8, Noble discloses the method of claim 6, further comprising: visualizing the aggregate complexity values by device class (For example, figure 4, the figure shows the complexity display which is a visualization of the aggregate complexity values).

As per claim 9, Noble discloses the method of claim 8, wherein the visualizing comprises: visualizing complexity values of layout components related to a selected device class [in a drill down portion] (For example, figure 4, the figure depicts a complexity evaluation hierarchy based on the selected (currently evaluated) user interface).

Noble does not expressly disclose a drill down portion.

However, Parker discloses a drill down portion (For example, figure 13 and 15, the display of the evaluation of the user interface is based on the selected device (WINDOWS or MAC operating environments (figure 13) or computer 1 or 2 (figure 14)). All devices tested allowing for a selection of one particular devices and then viewing results (i.e. drill down)).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by noble and extend it to multiple test multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

As per claim 10, Noble discloses a computer system having at least one computing device configured to run an integrated development environment that includes a complexity indicator according to claim 1 (For example, page 213 section 2.1, few design tools also display semantic information about the relationship between the interface and the rest of the program. For example, IBM's VisualAge is one such program which is an integrated development environment (IDE)).

the complexity indicator comprising: a library having complexity evaluation functions to determine complexity values of layout components of the respective layout component hierarchies, where each complexity evaluation function is specific to the layout component and [a device class] to which it is applied the complexity values being numerical values (For

example, page 213-215 sections 2.2-3.3, a suite (library) of metrics (complexity evaluation functions) are used to determine the complexity values of a corresponding user interface. The complexity evolution functions are associated with each of the components to which the functions are applied (see also figures 1-4). The functions evaluate to numerical values).

an aggregator to aggregate the complexity values into a [single complexity value for each device class] according to the corresponding layout component hierarchy of the respective [device class specific] representation (For example, figure 4, the results are aggregated together to show the relationships between the components (hierarchy) as well as the complexity of the interface layout. The display pertains to the specific user interface currently under observation but is not limited to just one user interface (i.e. the tool may execute multiple times with each resulting execution associated with a different UI)).

Noble does not expressly disclose device class specific representation.

However, Parker discloses device class specific representation (For example, figure 13 and 15, each operating environment and computer has a separate user interface design that is unique to that respective operating environment or computer).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble and extend it to multiple test multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

Noble and Parker do not expressly disclose an aggregator to aggregate the complexity values into a single complexity value for each [device class] and a complexity display to visually present an aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value.

However, Comber discloses an aggregator to aggregate the complexity values into a single complexity value for each [device class], a complexity display to visually present an aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value (For example, page 217, table 2 shows the aggregated complexity value for each GUI as a numerical value).

Noble, Parker and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system that determines complexity of each user interface, which executable on a different device class, numerically as described by Noble and Parker extend it to define complexity in terms of numerical, graph-able values per user interface as taught by Comber because it would provide with an efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

As per claim 12, Noble discloses a non-transitory machine-readable storage medium storing instruction which when executed by at least one processor provides a method comprising (For example, the software according to the invention executes on a 64M Sparc 4 processor. Processors are coupled to memory (machine-readable medium) that queue up

instructions that enable the processor to execute the instructions in a capable order to produce results).

determining complexity values of layout components of the [device class] specific representations by applying complexity evaluation functions that are specific to respective layout components [and the respective device class to which each complexity evaluation function is applied], the complexity values being numerical values (For example, pages 214-215 section 3-3.4, a suite (library) of metrics (functions) applied to different components of the GUI in determine complexity. Numerical values are obtained from the different metrics).

aggregating, using one or more processors, the complexity values [into a single complexity value for each device class] according to a corresponding layout component hierarchy of the respective device class specific representation (For example, figure 4, the results are aggregated together to show the relationships between the components (hierarchy) as well as the complexity of the interface layout. The display pertains to the specific user interface currently under observation but is not limited to just one user interface (i.e. the tool may execute multiple times with each resulting execution associated with a different UI); For example, page 217 section 4.4, the interactive visualization allows for the aggregation of the different metrics to allow for display in a single screen).

Noble does not expressly disclose the respective device class to which each complexity evaluation function is applied and receiving device class specific representations of the user interface, each device class specific representation referring a respective device class

However, Parker discloses the respective device class to which each complexity evaluation function is applied and receiving device class specific representations of the user

interface, each device class specific representation referring a respective device class (For example, figure 13 and 15, each operating environment and computer has a separate user interface design that is unique to that respective operating environment or computer).

Noble and Parker are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble and extend it to multiple test multiple user interfaces for different devices as taught by Parker because it would provide for the efficient means for simplifying the ease of testing user interfaces designed for multiple platforms (see Parker column 2 line 26-56).

Noble and Parker do not expressly disclose aggregating, using one or more processors, the complexity values into a single complexity value and visually presenting an aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value.

However, Comber discloses aggregating, using one or more processors, the complexity values into a single complexity value and visually presenting an aggregated complexity value for each [device class], the aggregated complexity value comprising a numerical value (For example, page 217, table 2 shows the aggregated complexity value for each GUI as a numerical value).

Noble, Parker and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system that determines complexity of each user interface, which executable on a different device class, numerically as described by Noble and Parker extend it to define complexity in terms of numerical, graph-able values per user interface as taught by Comber because it would provide with an efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

As per claim 13, Noble disclose the complexity indicator of claim 1, wherein the aggregator is to aggregate by propagating the complexity values of child nodes in the layout component hierarchy to a parent node (For example, figure 3, the two different Print tasks (parent nodes) complexity is shown in figures 3a and 3b. The tasks with the longer lines indicate higher complexity. The longer lines additionally indicate the complexity of the child nodes (subsequent buttons required for the task). The sum total of the child nodes, aggregated, indicate the complexity of the task. Once the buttons are rearranged the task is simpler (figure 3b)).

As per claim 14, Noble discloses the complexity indicator of claim 13, wherein the aggregator is to overrule one or more of the propagating complexity values with a higher complexity value calculated from the parent node (For example, figure 3, the two different Print tasks (parent nodes) complexity is shown in figures 3a and 3b. The tasks with the longer lines indicate higher complexity. The longer lines additionally indicate the complexity of the child nodes (subsequent buttons required for the task). The sum total of the child nodes, aggregated and thereby overruled, indicate the complexity of the task. Once the buttons are rearranged the

task is simpler (figure 3b). The sum total is an indication that the task's complexity is a sum of the complexity of the subsequent parts that make up the task and not any one single part).

As per claim 15, Noble discloses the complexity indicator of claim 1, wherein the complexity values comprise numerical values (For example, page 214 section 3, the metric visualization allows for the calculation (numerically) of all complexity values. Task concordance is computed as the rank order correlation, layout uniformity, function is shown in section 3.2 and visual coherence's function is shown in section 3.3).

As per claim 16, Noble does not expressly disclose the complexity indicator of claim 1, wherein the complexity display is to visually present the aggregated complexity value using a graphical bar.

However, Comber discloses wherein the complexity display is to visually present the aggregated complexity value using a graphical bar (For example, figure 3 and figure 8, the graph in figure 8 is a function of application ID and complexity. The graph uses a graphical bar to connect the complexities in order to show relative complexities between GUI's).

Noble and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble extend it to define complexity in terms of numerical, graph-able values as taught by Comber because it would provide for the

efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

As per claim 17, Noble disclose the method of claim 6, wherein the aggregating comprises propagating the complexity values of child nodes in the layout component hierarchy to a parent node (For example, figure 3, the two different Print tasks (parent nodes) complexity is shown in figures 3a and 3b. The tasks with the longer lines indicate higher complexity. The longer lines additionally indicate the complexity of the child nodes (subsequent buttons required for the task). The sum total of the child nodes, aggregated, indicate the complexity of the task. Once the buttons are rearranged the task is simpler (figure 3b)).

As per claim 18, Noble discloses the method of claim 17, further comprising overruling one or more of the propagating complexity values with a higher complexity value calculated from the parent node (For example, figure 3, the two different Print tasks (parent nodes) complexity is shown in figures 3a and 3b. The tasks with the longer lines indicate higher complexity. The longer lines additionally indicate the complexity of the child nodes (subsequent buttons required for the task). The sum total of the child nodes, aggregated and thereby overruled, indicate the complexity of the task. Once the buttons are rearranged the task is simpler (figure 3b). The sum total is an indication that the task's complexity is a sum of the complexity of the subsequent parts that make up the task and not any one single part).

As per claim 19, Noble discloses the method of claim 6, wherein the complexity values comprise numerical values (For example, page 214 section 3, the metric visualization allows for the calculation (numerically) of all complexity values. Task concordance is computed as the rank order correlation, layout uniformity, function is shown in section 3.2 and visual coherence's function is shown in section 3.3).

As per claim 20, Noble does not expressly disclose the method of claim 6, visually presenting the aggregated complexity value using a graphical bar.

However, Comber discloses wherein the complexity display is to visually present the aggregated complexity value using a graphical bar (For example, figure 3 and figure 8, the graph in figure 8 is a function of application ID and complexity. The graph uses a graphical bar to connect the complexities in order to show relative complexities between GUI's).

Noble and Comber are analogous art because they are from the same field of endeavor of testing user interfaces.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify the user interface testing system as described by Noble extend it to define complexity in terms of numerical, graph-able values as taught by Comber because it would provide for the efficient means for determining successful screen design by evaluating complexity (see Comber page 209 introduction).

Conclusion

9. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Samuel Hayim whose telephone number is (571) 270-3370. The examiner can normally be reached on Monday to Friday 8:30 AM to 5:00 PM.

If attempts to reach the above noted Examiner by telephone are unsuccessful, the Examiner's supervisor, Tuan Dam, can be reached at the following telephone number: (571) 272-3695.

The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/SAMUEL HAYIM/
Examiner, Art Unit 2192

/Thuy Dao/
Primary Examiner, Art Unit 2192